yocto PROJECT | THE LINUX FOUNDATION

Hands-On Kernel Lab: Introduction to linux-yocto, kernel config fragments and common workflow patterns

Tim Orling, Konsulko Group

Yocto Project Summit *Virtual*, 2022.05

Konsulko Group

- Services company specializing in Embedded Linux and Open Source Software
- Hardware/software build, design, development, and training services
- Based in San Jose, CA with an engineering presence worldwide
- https://konsulko.com/

Abstract

The Linux kernel is a key component of your board support package (BSP). In this session, we will discuss various practical ways of building the Linux kernel in the Yocto Project. We will cover building a traditional git tree and defconfig, an out-of-tree kernel module, a linux-yocto based kernel, adding kernel fragments for additional functionality and other common workflow patterns.

This session will be a combination of a talk and hands-on labs.

Description

The linux-yocto workflow is a powerful and flexible way to provide a consistent kernel experience across many platforms. But the use of yocto-kernel-cache metadata (a structured tree of kernel fragments) and the linux-yocto git repository often confuses newcomers to the Yocto Project. Many traditional developers prefer to use “a git tree and a defconfig” to build their kernel, so we will also cover this use case. Individual platforms will also differ in the use of bootloader, device tree and other details that involve where the kernel is installed and how it is booted. We will give hands-on practical examples of these use cases to help you on your journey to creating and working with well-behaved Yocto Project BSP layers.

Previous Hands-On Kernel Presentations

- Yocto Project Dev Day Virtual 2020 #3: Yocto Project Kernel Lab, Hands-On, Part 1 -- Trevor Woerner
- Yocto Project Dev Day Virtual 2020 #3: Yocto Project Kernel Lab, Hands-On, Part 2 -- Trevor Woerner
- Live Coding with Yocto Project #6: kernel handling and development -- Josef Holtzmayr (The Yocto Jester)
- Live Coding with Yocto Project #13: Building an out of tree kernel module -- Josef Holtzmayr (The Yocto Jester)
- “linux-yocto reference kernel maintenance and kernel workflows” -- Bruce Ashfield
- Working with the Linux Kernel in the Yocto Project -- Sean Hudson
- **Not an exhaustive list, there are more.**

Agenda

- **Why linux-yocto?**
- **Why kernel config fragments?**
  - yocto-kernel-cache aka KMETA
- **“Traditional” kernel developer workflows**
  - mainline kernel and defconfig
  - local git tree and defconfig
  - patch series with quilt
- **Yocto Project kernel best practices**
  - linux-yocto with config fragments and patches

Agenda

- **Hands-on Lab Exercises**
  - Lab #1: Mainline kernel and menuconfig
  - Lab #2: linux-yocto with config fragments and patches
  - Lab #3: Custom Kernel recipe
  - Lab #4: Custom Kernel recipe with local git tree

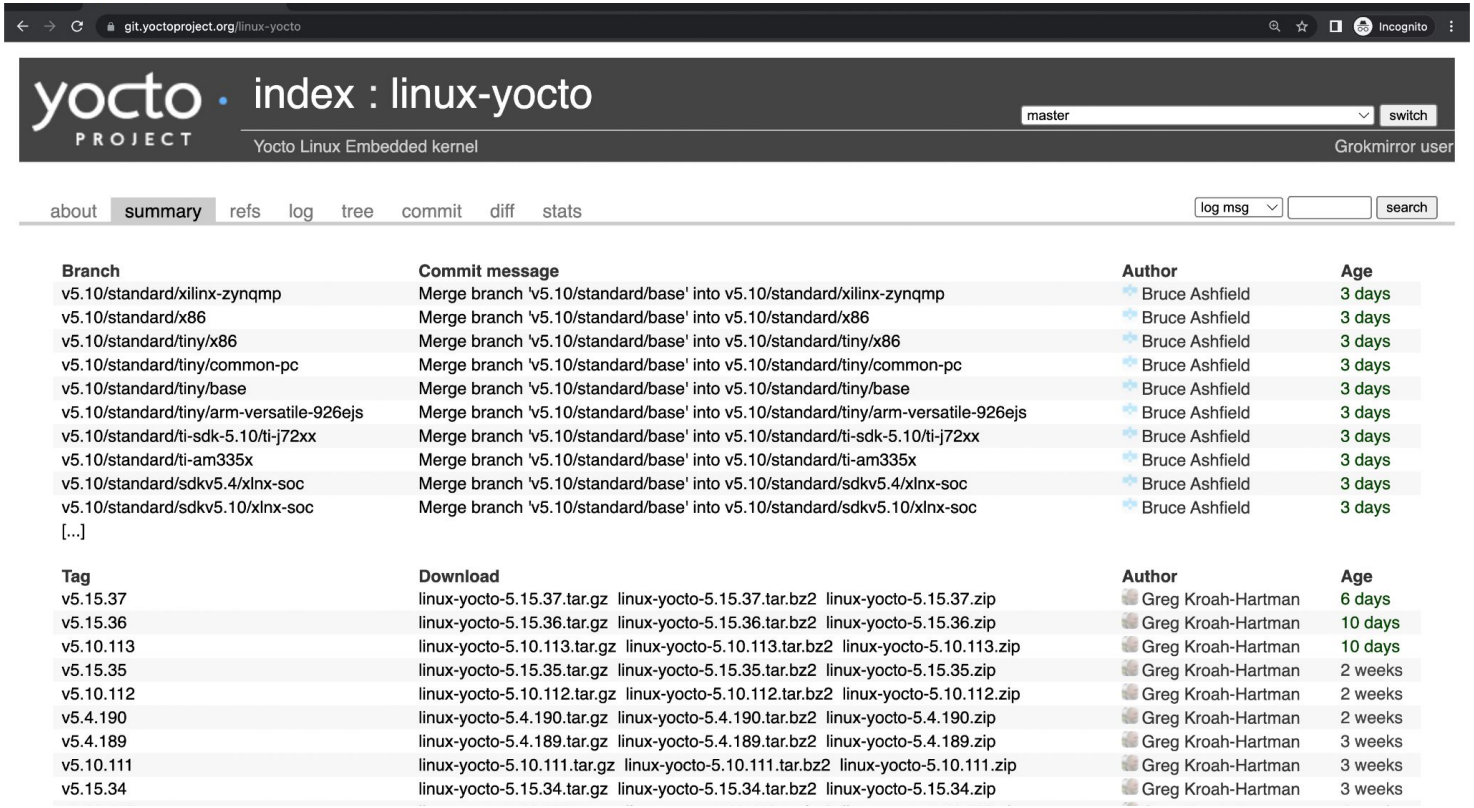
Why linux-yocto?

Why linux-yocto?

- Curated git tree
- Branches for SoCs and kernel versions
- Tested on supported SoCs
- Provides tooling to help protect you from bad configurations
- Provides tooling to help you maintain multiple kernel versions for multiple SoCs/MACHINES/boards.

What is linux-yocto?

https://git.yoctoproject.org/linux-yocto/



git.yoctoproject.org/linux-yocto

Incognito

yocto PROJECT

index : linux-yocto

Yocto Linux Embedded kernel

master switch

Grokmirror user

about summary refs log tree commit diff stats

log msg search

Branch	Commit message	Author	Age
v5.10/standard/xilinx-zynqmp	Merge branch 'v5.10/standard/base' into v5.10/standard/xilinx-zynqmp	Bruce Ashfield	3 days
v5.10/standard/x86	Merge branch 'v5.10/standard/base' into v5.10/standard/x86	Bruce Ashfield	3 days
v5.10/standard/tiny/x86	Merge branch 'v5.10/standard/base' into v5.10/standard/tiny/x86	Bruce Ashfield	3 days
v5.10/standard/tiny/common-pc	Merge branch 'v5.10/standard/base' into v5.10/standard/tiny/common-pc	Bruce Ashfield	3 days
v5.10/standard/tiny/base	Merge branch 'v5.10/standard/base' into v5.10/standard/tiny/base	Bruce Ashfield	3 days
v5.10/standard/tiny/arm-versatile-926ejs	Merge branch 'v5.10/standard/base' into v5.10/standard/tiny/arm-versatile-926ejs	Bruce Ashfield	3 days
v5.10/standard/ti-sdk-5.10/ti-j72xx	Merge branch 'v5.10/standard/base' into v5.10/standard/ti-sdk-5.10/ti-j72xx	Bruce Ashfield	3 days
v5.10/standard/ti-am335x	Merge branch 'v5.10/standard/base' into v5.10/standard/ti-am335x	Bruce Ashfield	3 days
v5.10/standard/sdkv5.4/xlnx-soc	Merge branch 'v5.10/standard/base' into v5.10/standard/sdkv5.4/xlnx-soc	Bruce Ashfield	3 days
v5.10/standard/sdkv5.10/xlnx-soc	Merge branch 'v5.10/standard/base' into v5.10/standard/sdkv5.10/xlnx-soc	Bruce Ashfield	3 days

[...]

Tag	Download	Author	Age
v5.15.37	linux-yocto-5.15.37.tar.gz linux-yocto-5.15.37.tar.bz2 linux-yocto-5.15.37.zip	Greg Kroah-Hartman	6 days
v5.15.36	linux-yocto-5.15.36.tar.gz linux-yocto-5.15.36.tar.bz2 linux-yocto-5.15.36.zip	Greg Kroah-Hartman	10 days
v5.10.113	linux-yocto-5.10.113.tar.gz linux-yocto-5.10.113.tar.bz2 linux-yocto-5.10.113.zip	Greg Kroah-Hartman	10 days
v5.15.35	linux-yocto-5.15.35.tar.gz linux-yocto-5.15.35.tar.bz2 linux-yocto-5.15.35.zip	Greg Kroah-Hartman	2 weeks
v5.10.112	linux-yocto-5.10.112.tar.gz linux-yocto-5.10.112.tar.bz2 linux-yocto-5.10.112.zip	Greg Kroah-Hartman	2 weeks
v5.4.190	linux-yocto-5.4.190.tar.gz linux-yocto-5.4.190.tar.bz2 linux-yocto-5.4.190.zip	Greg Kroah-Hartman	2 weeks
v5.4.189	linux-yocto-5.4.189.tar.gz linux-yocto-5.4.189.tar.bz2 linux-yocto-5.4.189.zip	Greg Kroah-Hartman	3 weeks
v5.10.111	linux-yocto-5.10.111.tar.gz linux-yocto-5.10.111.tar.bz2 linux-yocto-5.10.111.zip	Greg Kroah-Hartman	3 weeks
v5.15.34	linux-yocto-5.15.34.tar.gz linux-yocto-5.15.34.tar.bz2 linux-yocto-5.15.34.zip	Greg Kroah-Hartman	3 weeks

What is linux-yocto?

- Curated kernel recipes

```
meta/recipes-kernel/linux/
├── kernel-devsrc.bb
├── linux-dummy
│   └── COPYING.GPL
├── linux-dummy.bb
├── linux-yocto-dev.bb
├── linux-yocto-rt_5.10.bb
├── linux-yocto-rt_5.15.bb
├── linux-yocto-tiny_5.10.bb
├── linux-yocto-tiny_5.15.bb
├── linux-yocto.inc
├── linux-yocto_5.10.bb
└── linux-yocto_5.15.bb
```

What is linux-yocto?

- Curated kernel bbclasses

```
meta/classes/
...
├── devicetree.bbclass
...
├── kernel-arch.bbclass
├── kernel-artifact-names.bbclass
├── kernel-devicetree.bbclass
├── kernel-fitimage.bbclass
├── kernel-grub.bbclass
├── kernel-module-split.bbclass
├── kernel-uboot.bbclass
├── kernel-uimage.bbclass
├── kernel-yocto.bbclass
├── kernel.bbclass
├── kernelsrc.bbclass
...
├── linux-dummy.bbclass
├── linux-kernel-base.bbclass
├── module-base.bbclass
├── module.bbclass
...
```

Why kernel fragments?

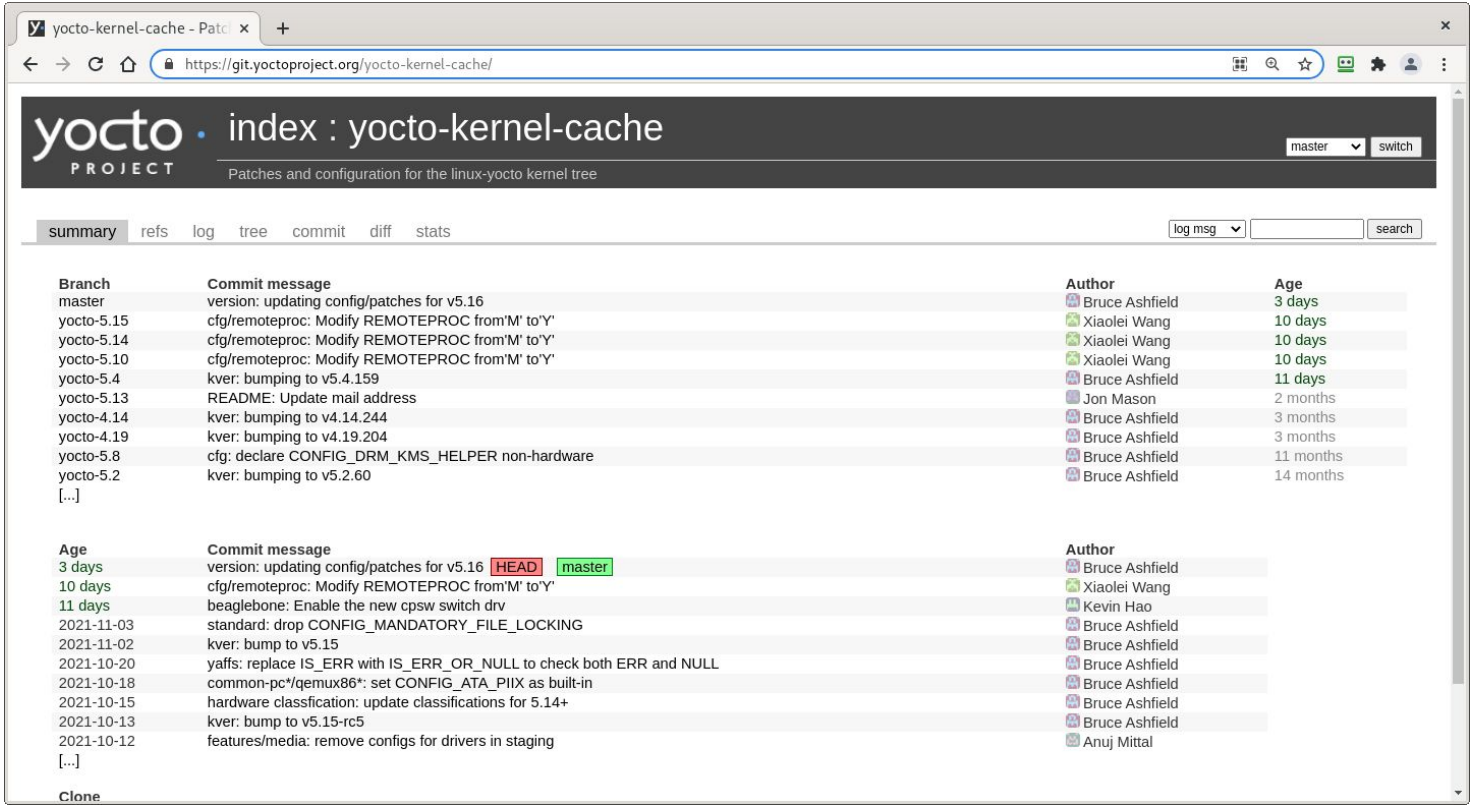
...and what is yocto-kernel-cache?

Why kernel fragments?

- Fully supported in the Linux kernel “It’s normal”
- Modular approach to configuring the kernel
- Allows base kernel configurations to be re-used
- Allows SoC-base configurations to be maintained for MANY SoCs

What is yocto-kernel-cache?

https://git.yoctoproject.org/yocto-kernel-cache/



Structure of yocto-kernel-cache

```
yocto-kernel-cache
├── 00-README
├── arch
├── backports
├── bsp
├── cfg
├── cgl
├── COPYING.GPLv2
├── COPYING.MIT
├── features
├── kern-features.rc
├── ktypes
├── kver
├── patches
├── scripts
├── small
└── staging
```

```
cfg
├── 8250.cfg
├── 8250.scc
...
├── fs
...
├── net
...
├── timer
...
```

```
features
├── 6lowpan
├── apparmor
├── aufs
├── bfq
├── blktrace
├── bluetooth
├── bpf
├── bsdjail
├── can
├── cgroups
├── ciphers
...
```

.cfg and .scc files

.cfg: The config fragments

```
$ cat cfg/8250.cfg
# SPDX-License-Identifier: MIT
CONFIG_TTY=y
CONFIG_SERIAL_8250=y
CONFIG_SERIAL_8250_CONSOLE=y
CONFIG_SERIAL_8250_PCI=y
CONFIG_SERIAL_8250_NR_UARTS=4
CONFIG_SERIAL_8250_RUNTIME_UARTS=4
CONFIG_SERIAL_CORE=y
CONFIG_SERIAL_CORE_CONSOLE=y
CONFIG_SERIAL_OF_PLATFORM=y
```

.scc: “Series Configuration Control” (metadata)

```
$ cat cfg/8250.scc
# SPDX-License-Identifier: MIT
define KFEATURE_DESCRIPTION "Enable
8250 serial support"
define KFEATURE_COMPATIBILITY board

kconf hardware 8250.cfg
```

“Traditional” kernel workflows

“Traditional” Kernel Developer Workflows

- mainline kernel and defconfig
  - “I just want to use a mainline kernel and a defconfig”
  - Commonly a vanilla tarball from kernel.org
  - Commonly a full “defconfig” edited with menuconfig.
  - We’ll use this workflow in Lab #1.

“Traditional” Kernel Developer Workflows

- local git tree and defconfig
  - “I just want to use a git tree and a defconfig”
  - Commonly a local git tree
  - Commonly a full “defconfig” edited with menuconfig
  - We’ll use this workflow in Lab #3

“Traditional” Kernel Developer Workflows

- patch series with quilt (and defconfig)
  - “I just want to use a quilt patch series and a defconfig”
  - Commonly (re)based on top of a vanilla mainline git tree
  - Commonly a full “defconfig” edited with menuconfig.
  - See Using Quilt in your Workflow
  - This will be a future Lab activity.

Yocto Project kernel best practices

linux-yocto with config fragments and patches

Yocto Project Kernel Best Practices

- Don’t create an “evil vendor kernel”
- Unless absolutely impossible, base your kernel on linux-yocto (with patches)
- Don’t use a full defconfig. Use kernel config fragments based on top of yocto-kernel-cache
- Don’t create an “evil vendor kernel”
- Don’t create an “evil vendor kernel”
- We’ll investigate this workflow in Labs #2 and #4.

Hands-on Lab Exercises

Setting up your session

- **You might need to switch to your user account**
  - su - <username>
- **You might need to install tmux (and you might want mosh)**
  - sudo apt install tmux mosh
- **You might want to return to where you left off**
  - tmux new -s kernel-lab
- **When you return (or your web console times out)**
  - tmux a -t kernel-lab

Preparing Your Environment

- **Cloning the layers**
  - cd ~
  - git clone --depth 1 --branch kirkstone https://git.yoctoproject.org/poky.git
  - git clone --depth 1 --branch kirkstone https://github.com/moto-timo/kernel-lab-layers.git

Setup your local.conf (if you are self-provisioning)

- `$ cd ~`
- `$ . poky/oe-init-build-env build-kernel/`
- `$ vim conf/local.conf`

- Add the following to setup the same environment as the slides

```
# Kernel Hands-on Lab machines
#MACHINE = "lab1-qemuarm64"
#MACHINE = "lab2-qemuarm64"
#MACHINE = "lab3-qemuarm64"
#MACHINE = "lab4-qemuarm64"

DL_DIR ?= "${HOME}/DOWNLOADS"
SSTATE_DIR ?= "${HOME}/SSTATE"
```

Hands-on Kernel Lab #1
Exercise #1

Working with kernel.org tarball and menuconfig

Lab #1

- Getting to know the environment
  - ~/kernel-lab-layers/meta-lab1-qemuarm64

```
meta-lab1-qemuarm64/
├── conf
│   ├── layer.conf
│   └── machine
│       └── lab1-qemuarm64.conf
└── recipes-kernel
    └── linux
        ├── linux-korg
        │   ├── arm64_defconfig
        │   ├── defconfig
        │   ├── qemuarm64_defconfig
        │   └── yocto-testmod.patch
        └── linux-korg_5.17.bb
```

Latest Stable “kernel.org” 5.17.5

linux-korg_5.17.bb

~/kernel-lab-layers/meta-lab1-qemuarm64/recipes-kernel/linux/linux-korg_5.17.bb

```
DESCRIPTION = "Mainline Linux Kernel"
SECTION = "kernel"
LICENSE = "GPL-2.0-only"

FILESEXTRAPATHS:prepend := "${THISDIR}/files:"

LIC_FILES_CHKSUM = "file://COPYING;md5=6bc538ed5bd9a7fc9398086aedcd7e46"

inherit kernel

SRC_URI = "${KERNELORG_MIRROR}/linux/kernel/v5.x/linux-${PV}.tar.xz;name=kernel \
           file://defconfig"

S = "${WORKDIR}/linux-${PV}"

#SRC_URI += "file://yocto-testmod.patch"

PV = "5.17.5"
SRC_URI[kernel.sha256sum] = "9bbcd185b94436f9c8fe977fa0e862f60d34003562327fcebb27c9fa342fe987"
```

machine/lab1-qemu.conf

~/yocto-kernel-lab-layers/meta-lab1-qemuarm64/conf/machine/lab1-qemuarm64.conf

```
#@TYPE: Machine
#@NAME: lab1-qemuarm64

#@DESCRIPTION: Machine configuration for lab1-qemuarm64 systems

PREFERRED_PROVIDER_virtual/kernel ?= "linux-korg"

PREFERRED_PROVIDER_virtual/xserver ?= "xserver-xorg"
PREFERRED_PROVIDER_virtual/libgl ?= "mesa"
PREFERRED_PROVIDER_virtual/libgles1 ?= "mesa"
PREFERRED_PROVIDER_virtual/libgles2 ?= "mesa"

require conf/machine/include/qemu.inc
require conf/machine/include/arm/armv8a/tune-cortexa57.inc

KERNEL_IMAGETYPE = "Image"

UBOOT_MACHINE ?= "qemu_arm64_defconfig"
...
```

Setup our build environment

- `$ cd ~`
- `$ . poky/oe-init-build-env build-kernel/`
- `$ vim conf/local.conf`
- uncomment `#MACHINE = “lab1-qemuarm64”`
- save and exit vim (`:wq`)
- `$ bitbake-layers add-layer ~/kernel-lab-layers/meta-lab1-qemuarm64`
- `$ cat conf/bblayers.conf`

```
BBLAYERS ?= " \
  /home/<user>/poky/meta \
  /home/<user>/poky/meta-poky \
  /home/<user>/poky/meta-yocto-bsp \
  /home/<user>/kernel-lab-layers/meta-lab1-qemuarm64 \
  "
```

Lab #1 -- Build and Boot the Image

- `bitbake core-image-base`
- `runqemu slirp nographic tmp/deploy/images/lab1-qemuarm64/Image-lab1-qemuarm64.bin tmp/deploy/images/lab1-qemuarm64/core-image-base-lab1-qemuarm64.ext4`
  - slirp: user space networking (no elevated privileges required)
  - nographic: run in console, do not launch a GUI window
  - path to kernel image
  - path to rootfs

kernel panic!

```
[    3.130236] No filesystem could mount root, tried:
[    3.130274]  vfat
[    3.130462]  btrfs
[    3.130545]
[    3.130896] Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block(253,0)
[    3.131415] CPU: 3 PID: 1 Comm: swapper/0 Not tainted 5.17.5 #1
[    3.131823] Hardware name: linux,dummy-virt (DT)
[    3.132313] Call trace:
[    3.132559]  dump_backtrace.part.0+0xc4/0xd0
[    3.132989]  show_stack+0x24/0x40
[    3.133343]  dump_stack_lvl+0x7c/0xa0
[    3.134023]  dump_stack+0x18/0x34
[    3.134194]  panic+0x168/0x32c
[    3.134344]  mount_block_root+0x224/0x240
[    3.134550]  mount_root+0x210/0x24c
[    3.134710]  prepare_namespace+0x140/0x180
[    3.134889]  kernel_init_freeable+0x268/0x29
[    3.135081]  kernel_init+0x34/0x140
[    3.135229]  ret_from_fork+0x10/0x20
[    3.135684] SMP: stopping secondary CPUs
[    3.136511] Kernel Offset: disabled
[    3.136692] CPU features: 0x44,00000342,00000842
[    3.137236] Memory Limit: 256 MB
[    3.137856] ---[ end Kernel panic - not syncing: VFS: Unable to mount root fs on
unknown-block(253,0) ]---
```

Where's ext4?

?

Ctrl-A X

Inspect the defconfig

```
grep -R EXT4 ~/kernel-lab-layers/meta-lab1-qemuarm64/recipes-kernel/linux/linux-korg/defconfig

# CONFIG_EXT4_FS is not set
```

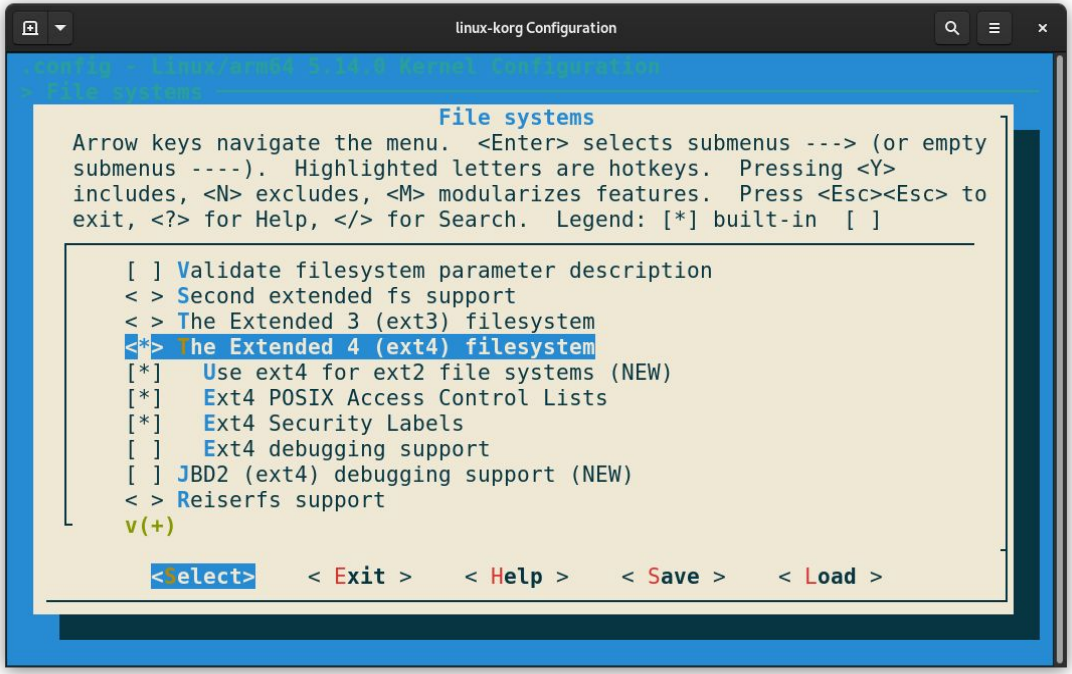
Lab #1

- **How to exit QEMU if your terminal becomes unresponsive**
  - Connect to your droplet with another terminal or the Web UI Console
  - # top
  - Look for “qemu-system-aar…” and note the PID
  - # kill -s SIGTERM <PID>

Lab #1 -- Configure the Kernel

```
PREFERRED_PROVIDER:virtual/kernel = "linux-korg"
```

- `bitbake -c menuconfig virtual/kernel`



Lab #1 -- Rebuild (only) the Kernel

```
PREFERRED_PROVIDER:virtual/kernel = "linux-korg"
```

- **`bitbake -c compile -f virtual/kernel`**
  - This forces the "**`do_compile`**" task to be rerun.
- **`bitbake -c deploy virtual/kernel`**
  - This deploys the kernel image and artifacts to the **`tmp/deploy/images/lab1-qemuarm64`** directory.

```
MACHINE = "lab1-qemuarm64"
```

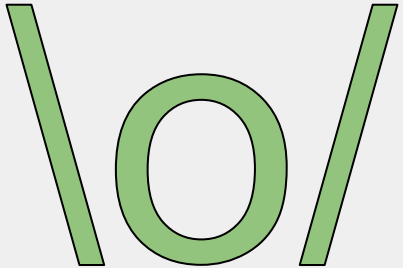
Lab #1 -- Boot the Image

- `runqemu slirp nographic tmp/deploy/images/lab1-qemuarm64/Image-lab1-qemuarm64.bin tmp/deploy/images/lab1-qemuarm64/core-image-base-lab1-qemuarm64.ext4`
  - slirp: user space networking (no elevated privileges required)
  - nographic: run in console, do not launch a GUI window
  - path to kernel image
  - path to rootfs

Success!

```
Poky (Yocto Project Reference Distro) 4.0 lab1-qemuarm64 /dev/ttyAMA0

lab1-qemuarm64 login: root
root@lab1-qemuarm64:~# uname -a
Linux lab1-qemuarm64 5.17.5 #1 SMP PREEMPT Wed Apr 27 12:41:17 UTC 2022 aarch64 GNU/Linux
root@lab1-qemuarm64:~# dmesg | grep EXT4
[    2.809670] EXT4-fs (vda): recovery complete
[    2.813628] EXT4-fs (vda): mounted filesystem with ordered data mode. Quota mode: disabled.
[    6.564263] EXT4-fs (vda): re-mounted. Quota mode: disabled.
```



diff .config defconfig

the .config that menuconfig saved (our build .config)

copy of our recipe's defconfig

```
$ diff tmp/work/lab1_qemuarm64-poky-linux/linux-korg/5.17.5-r0/build/.config
tmp/work/lab1_qemuarm64-poky-linux/linux-korg/5.17.5-r0/defconfig
3507,3514c3507
< CONFIG_EXT4_FS=y
< CONFIG_EXT4_USE_FOR_EXT2=y
< CONFIG_EXT4_FS_POSIX_ACL=y
< CONFIG_EXT4_FS_SECURITY=y
< # CONFIG_EXT4_DEBUG is not set
< CONFIG_JBD2=y
< # CONFIG_JBD2_DEBUG is not set
< CONFIG_FS_MBCACHE=y
---
> # CONFIG_EXT4_FS is not set
3535d3527
< CONFIG_FS_ENCRYPTION_ALGS=y
```

To share the change (make it ‘permanent’)

```
$ cp
tmp/work/lab1_qemuarm64-poky-linux/linux-korg/5.17.5-r0/build/.config
~/kernel-lab-layers/meta-lab1-qemuarm64/recipes-kernel/linux/linux-korg/
defconfig
```

Hands-on Kernel Lab #1
Exercise #2

Patch the kernel to add a custom kernel module

linux-korg_5.17.bb: uncomment yocto-testmod.patch

~/kernel-lab-layers/meta-lab1-qemuarm64/recipes-kernel/linux/linux-korg_5.17.bb

```
DESCRIPTION = "Mainline Linux Kernel"
SECTION = "kernel"
LICENSE = "GPL-2.0-only"

FILESEXTRAPATHS:prepend := "${THISDIR}/files:"

LIC_FILES_CHKSUM = "file://COPYING;md5=6bc538ed5bd9a7fc9398086aedcd7e46"

inherit kernel

SRC_URI = "${KERNELORG_MIRROR}/linux/kernel/v5.x/linux-${PV}.tar.xz;name=kernel \
           file://defconfig"

S = "${WORKDIR}/linux-${PV}"

SRC_URI += "file://yocto-testmod.patch"

PV = "5.17.5"
SRC_URI[kernel.sha256sum] = "9bbcd185b94436f9c8fe977fa0e862f60d34003562327fcebb27c9fa342fe987"
```

yocto-testmod.patch

~/kernel-lab-layers/meta-lab1-qemuarm64/recipes-kernel/linux/linux-korg/yocto-testmod.patch

- Add module to Kconfig

```
+config YOCTO_TESTMOD
+    tristate "Yocto Test Driver"
+    help
+      This driver provides a silly message for testing Yocto.
+
```

- Add Makefile
- Add module init code (pr_info)

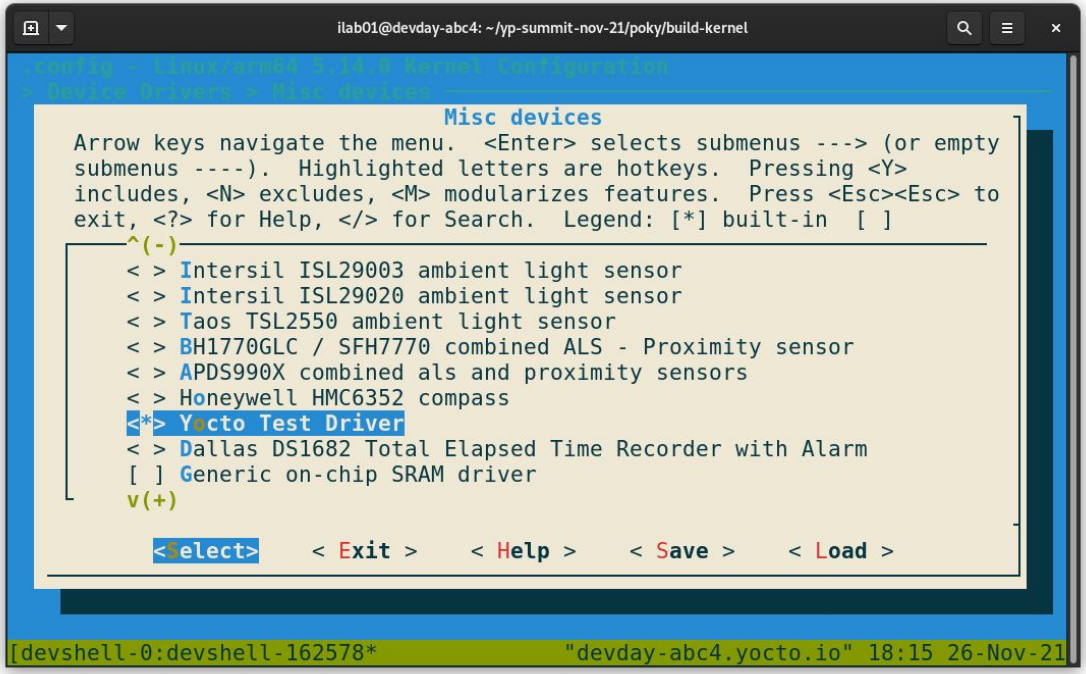
Modern version of printk

```
+#include <linux/module.h>
+
+static int __init yocto_testmod_init(void)
+{
+    pr_info("Kilroy was here!");
+
+    return 0;
+}
```

Lab #1 -- Configure the Kernel

PREFERRED_PROVIDER:virtual/kernel = “linux-korg”

- `bitbake -c menuconfig virtual/kernel`



Lab #1 -- Rebuild (only) the Kernel

- `bitbake -c compile -f virtual/kernel`
  - This forces the "`do_compile`" task to be rerun.
  - This is actually now optional since the recipe is already "tainted".
- `bitbake -c deploy virtual/kernel`
  - This deploys the kernel image and artifacts to the `tmp/deploy/images/lab1-qemuarm64` directory.

Lab #1 -- Boot the Image

"Up" arrow to reuse the command (it hasn't changed) or "Ctrl-R" and start typing for bash completion.

- `runqemu slirp nographic tmp/deploy/images/lab1-qemuarm64/Image-lab1-qemuarm64.bin tmp/deploy/images/lab1-qemuarm64/core-image-base-lab1-qemuarm64.ext4`
  - slirp: user space networking (no elevated privileges required)
  - nographic: run in console, do not launch a GUI window
  - path to kernel image
  - path to rootfs

Success!

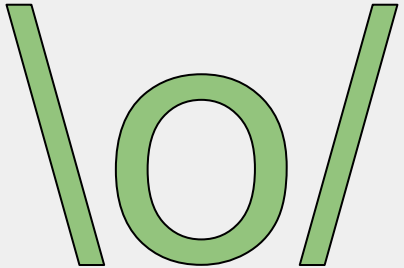
If you get a kernel panic, also make and save the EXT4 File System changes we made earlier.

```
[    2.918091] EXT4-fs (vda): recovery complete
[    2.921406] EXT4-fs (vda): mounted filesystem with ordered data mode. Quota mode: disabled.

...

Poky (Yocto Project Reference Distro) 4.0 lab1-qemuarm64 /dev/ttyAMA0

lab1-qemuarm64 login: root
root@lab1-qemuarm64:~# dmesg | grep Kilroy
[    2.450176] Kilroy was here!
root@lab1-qemuarm64:~#
```



To share the change (make it ‘permanent’)

```
$ cp
tmp/work/lab1_qemuarm64-poky-linux/linux-korg/5.17.5-r0/build/.config
~/kernel-lab-layers/meta-lab1-qemuarm64/recipes-kernel/linux/linux-korg/
defconfig
```

Lab#1 Complete!

Hands-on Kernel Lab #2
Exercise #1

Working with linux-yocto and config fragments

Lab #2

- Getting to know the environment
  - ~/kernel-lab-layers/meta-lab2-qemuarm64

```
meta-lab2-qemuarm64/
├── conf
│   ├── layer.conf
│   └── machine
│       └── lab2-qemuarm64.conf
└── recipes-kernel
    └── linux
        ├── files
        │   ├── lab2.cfg
        │   ├── mtd-block.cfg
        │   └── yocto-testmod.patch
        ├── linux-yocto_5.10.bbappend
        └── linux-yocto_5.15.bbappend
```

machine/lab2-qemu.conf

~/kernel-lab-layers/meta-lab2-qemuarm64/conf/machine/lab2-qemuarm64.conf

```
#@TYPE: Machine
#@NAME: lab2-qemuarm64

#@DESCRIPTION: Machine configuration for lab2-qemuarm64 systems

PREFERRED_PROVIDER_virtual/kernel ?= "linux-yocto"
PREFERRED_VERSION_linux-yocto ?= "5.15%"
#PREFERRED_VERSION_linux-yocto ?= "5.10%"


require conf/machine/include/qemu.inc
require conf/machine/include/arm/armv8a/tune-cortexa57.inc

KERNEL_IMAGETYPE = "Image"

UBOOT_MACHINE ?= "qemu_arm64_defconfig"
...
```

linux-yocto_5.15.bbappend

~/kernel-lab-layers/meta-lab2-qemuarm64/recipes-kernel/linux/linux-yocto_5.15.bbappend

```
FILESEXTRAPATHS:prepend := "${THISDIR}/files:"

COMPATIBLE_MACHINE:lab2-qemuarm64 = "lab2-qemuarm64"

KBRANCH:lab2-qemuarm64 = "v5.15/standard/base"
KMACHINE:lab2-qemuarm64  = "qemuarm64"

KERNEL_FEATURES:append:lab2-qemuarm64 = " cfg/smp.scc"

#SRC_URI += "file://yocto-testmod.patch"
#SRC_URI += "file://lab2.cfg"
```

Setup our build environment

We're already in this environment from Lab#1

- `$ cd ~`
- `$ . poky/oe-init-build-env build-kernel/`
- `$ vim conf/local.conf`
- re-comment `#MACHINE = "lab1-qemuarm64"`
- uncomment `#MACHINE = "lab2-qemuarm64"`
- save and exit vim (`:wq`)
- `$ bitbake-layers remove-layer ~/kernel-lab-layers/meta-lab1-qemuarm64`
- `$ bitbake-layers add-layer ~/kernel-lab-layers/meta-lab2-qemuarm64`
- `$ cat conf/bblayers.conf`

```
    /home/<user>/kernel-lab-layers/meta-lab2-qemuarm64 \
```

Lab #2 -- Build and Boot the Image

kernel compilation is ~7.75 minutes
kernel deploy is ~2 minutes

- `bitbake core-image-base` (~1.5 minutes: through the magic of sstate)
- `runqemu slirp nographic tmp/deploy/images/lab2-qemuarm64/Image-lab2-qemuarm64.bin tmp/deploy/images/lab2-qemuarm64/core-image-base-lab2-qemuarm64.ext4`
  - slirp: user space networking (no elevated privileges required)
  - nographic: run in console, do not launch a GUI window
  - path to kernel image
  - path to rootfs

First boot

```
Poky (Yocto Project Reference Distro) 4.0 lab2-qemuarm64 /dev/ttyAMA0

lab2-qemuarm64 login: root
root@lab2-qemuarm64:~# uname -a
Linux lab2-qemuarm64 5.15.32-yocto-standard #1 SMP PREEMPT Mon Mar 28 19:23:07 UTC 2022 aarch64
GNU/Linux
root@lab2-qemuarm64:~#
```

yocto-testmod.patch

~/kernel-lab-layers/meta-lab2-qemuarm64/recipes-kernel/linux/files/yocto-testmod.patch

- Add module to Kconfig

```
+config YOCTO_TESTMOD
+    tristate "Yocto Test Driver"
+    help
+      This driver provides a silly message for testing Yocto.
+
```

- Add Makefile
- Add module init code (pr_info)

```
+#include <linux/module.h>
+
+static int __init yocto_testmod_init(void)
+{
+    pr_info("Krillroy swam here!");
+
+    return 0;
+}
```

lab2.cfg

~/kernel-lab-layers/meta-lab2-qemuarm64/recipes-kernel/linux/files/lab2.cfg

- Enable our test module

```
# Enable the testmod
CONFIG_YOCTO_TESTMOD=y
```

linux-yocto_5.15.bbappend: add patch and config

~/kernel-lab-layers/meta-lab2-qemuarm64/recipes-kernel/linux/linux-yocto_5.15.bbappend

```
FILESEXTRAPATHS:prepend := "${THISDIR}/files:"

COMPATIBLE_MACHINE:lab2-qemuarm64 = "lab2-qemuarm64"

KBRANCH:lab2-qemuarm64 = "v5.15/standard/base"
KMACHINE:lab2-qemuarm64  = "qemuarm64"

KERNEL_FEATURES:append:lab2-qemuarm64 = " cfg/smp.scc"

SRC_URI += "file://yocto-testmod.patch"
SRC_URI += "file://lab2.cfg"
```

Uncomment these two SRC_URI lines

Lab #2 -- Rebuild (only) the Kernel

```
PREFERRED_PROVIDER:virtual/kernel = "linux-yocto"
```

- **`bitbake -c compile -f virtual/kernel`**
  - This forces the "**`do_compile`**" task to be rerun.
- **`bitbake -c deploy virtual/kernel`**
  - This deploys the kernel image and artifacts to the **`tmp/deploy/images/lab2-qemuarm64`** directory.

```
MACHINE = "lab2-qemuarm64"
```

Lab #2 -- Boot the Image

"Up" arrow to reuse the command (it hasn't changed) or "Ctrl-R" and start typing for bash completion.

- `runqemu slirp nographic tmp/deploy/images/lab2-qemuarm64/Image-lab2-qemuarm64.bin tmp/deploy/images/lab2-qemuarm64/core-image-base-lab2-qemuarm64.ext4`
  - slirp: user space networking (no elevated privileges required)
  - nographic: run in console, do not launch a GUI window
  - path to kernel image
  - path to rootfs

Success!

```
[    6.580570] EXT4-fs (vda): re-mounted. Opts: (null). Quota mode: disabled.

   ...

Poky (Yocto Project Reference Distro) 4.0 lab2-qemuarm64 /dev/ttyAMA0

lab2-qemuarm64 login: root
root@lab2-qemuarm64:~# dmesg | grep Krillroy
[    2.870975] Krillroy swam here!
root@lab2-qemuarm64:~#
```

Hands-on Kernel Lab #2
Exercise #2

Modify the Kernel to Make Use of an LTS Kernel Option

machine/lab2-qemu.conf: switch to 5.10 LTS kernel

~/kernel-lab-layers/meta-lab2-qemuarm64/conf/machine/lab2-qemuarm64.conf

```
#@TYPE: Machine
#@NAME: lab2-qemuarm64

#@DESCRIPTION: Machine configuration for lab2-qemuarm64 systems

PREFERRED_PROVIDER_virtual/kernel ?= "linux-yocto"
#PREFERRED_VERSION_linux-yocto ?= "5.15%"
PREFERRED_VERSION_linux-yocto ?= "5.10%"


require conf/machine/include/qemu.inc
require conf/machine/include/arm/armv8a/tune-cortexa57.inc

KERNEL_IMAGETYPE = "Image"

UBOOT_MACHINE ?= "qemu_arm64_defconfig"
...
```

linux-yocto_5.10.bbappend (an LTS kernel)

~/kernel-lab-layers/meta-lab2-qemuarm64/recipes-kernel/linux/linux-yocto_5.10.bbappend

```
FILESEXTRAPATHS:prepend := "${THISDIR}/files:"

COMPATIBLE_MACHINE:lab2-qemuarm64 = "lab2-qemuarm64"

KBRANCH:lab2-qemuarm64 = "v5.10/standard/base"
KMACHINE:lab2-qemuarm64  = "qemuarm64"

KERNEL_FEATURES:append:lab2-qemuarm64 = " cfg/smp.scc"

#SRC_URI += "file://mtd-block.cfg"

#SRC_URI += "file://yocto-testmod.patch"
#SRC_URI += "file://lab2.cfg"
```

Lab #2 -- Rebuild (only) the Kernel

```
PREFERRED_PROVIDER:virtual/kernel = "linux-yocto"
```

- **`bitbake -c compile -f virtual/kernel`**
  - This forces the "**`do_compile`**" task to be rerun.
  - This time, **5.10** kernel will be built (PREFERRED_VERSION)
- **`bitbake -c deploy virtual/kernel`**
  - This deploys the kernel image and artifacts to the **`tmp/deploy/images/lab2-qemuarm64`** directory.

```
MACHINE = "lab2-qemuarm64"
```

Lab #2 -- Boot the Image

"Up" arrow to reuse the command (it hasn't changed) or "Ctrl-R" and start typing for bash completion.

- `runqemu slirp nographic tmp/deploy/images/lab2-qemuarm64/Image-lab2-qemuarm64.bin tmp/deploy/images/lab2-qemuarm64/core-image-base-lab2-qemuarm64.ext4`
  - slirp: user space networking (no elevated privileges required)
  - nographic: run in console, do not launch a GUI window
  - path to kernel image
  - path to rootfs

Boot with LTS Kernel

```
Poky (Yocto Project Reference Distro) 4.0 lab2-qemuarm64 /dev/ttyAMA0

lab2-qemuarm64 login: root
root@lab2-qemuarm64:~# uname -a
Linux lab2-qemuarm64 5.10.109-yocto-standard #1 SMP PREEMPT Mon Mar 28 19:14:33 UTC 2022 aarch64
GNU/Linux
root@lab2-qemuarm64:~#
```

mtd-block.cfg

~/kernel-lab-layers/meta-lab2-qemuarm64/recipes-kernel/linux/files/mtd-block.cfg

- Add Memory Technology Device block device support

```
# Enable MTD BLOCK
CONFIG_MTD=y
CONFIG_MTD_BLOCK=y
```

linux-yocto_5.10.bbappend: add mtd-block fragment

~/kernel-lab-layers/meta-lab2-qemuarm64/recipes-kernel/linux/linux-yocto_5.10.bbappend

```
FILESEXTRAPATHS:prepend := "${THISDIR}/files:"

COMPATIBLE_MACHINE:lab2-qemuarm64 = "lab2-qemuarm64"

KBRANCH:lab2-qemuarm64 = "v5.10/standard/base"
KMACHINE:lab2-qemuarm64  = "qemuarm64"

KERNEL_FEATURES:append:lab2-qemuarm64 = " cfg/smp.scc"

SRC_URI += "file://mtd-block.cfg"

#SRC_URI += "file://yocto-testmod.patch"
#SRC_URI += "file://lab2.cfg"
```

Lab #2 -- Rebuild (only) the Kernel

- `bitbake -c compile -f virtual/kernel`
  - This forces the "`do_compile`" task to be rerun.
  - This is actually now optional since the recipe is already "tainted".
- `bitbake -c deploy virtual/kernel`
  - This deploys the kernel image and artifacts to the `tmp/deploy/images/lab2-qemuarm64` directory.

Examine the .config

linux-yocto build directory is linux-<MACHINE>-standard-build

```
$ vim
tmp/work/lab2_qemuarm64-poky-linux/linux-yocto/5.10.109+gitAUTOINC+2278ed571c_d2f7a595bf-r0/linux
-lab2_qemuarm64-standard-build/.config

CONFIG_MTD=y
# CONFIG_MTD_TESTS is not set

...

#
# User Modules And Translation Layers
#
CONFIG_MTD_BLKDEVS=y
CONFIG_MTD_BLOCK=y
```

KMETA hash

KBRANCH hash

Search with /CONFIG_MTD=y

Lab#2 Complete!

Hands-on Kernel Lab #3
Exercise #1

Working with a Custom Kernel Recipe

(When in doubt) Choosing a Kernel

https://www.kernel.org/

Lab #3

- Getting to know the environment
  - ~/kernel-lab-layers/meta-lab3-qemuarm64

```
meta-lab3-qemuarm64/
├── conf
│   ├── layer.conf
│   └── machine
│       └── lab3-qemuarm64.conf
└── recipes-kernel
    └── linux
        ├── linux-yocto-custom
        │   ├── arm64_defconfig
        │   ├── defconfig
        │   ├── lab3.cfg
        │   ├── qemuarm64_defconfig
        │   └── yocto-testmod.patch
        └── linux-yocto-custom_git.bb
```

machine/lab3-qemu.conf

~/kernel-lab-layers/meta-lab3-qemuarm64/conf/machine/lab3-qemuarm64.conf

```
#@TYPE: Machine
#@NAME: lab3-qemuarm64

#@DESCRIPTION: Machine configuration for lab3-qemuarm64 systems

PREFERRED_PROVIDER_virtual/kernel ?= "linux-yocto-custom"

require conf/machine/include/qemu.inc
require conf/machine/include/arm/armv8a/tune-cortexa57.inc

KERNEL_IMAGETYPE = "Image"

UBOOT_MACHINE ?= "qemu_arm64_defconfig"

...
```

linux-yocto-custom_git.bb

~/kernel-lab-layers/meta-lab3-qemuarm64/recipes-kernel/linux/linux-yocto-custom_git.bb

```
inherit kernel
require recipes-kernel/linux/linux-yocto.inc

# KBRANCH is the branch the used for the git clone. In this case the tip of 5.15 stable
KBRANCH = "linux-5.17.y"

SRC_URI:lab3-qemuarm64 =
"git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git;protocol=git;nocheckout=1;
branch=${KBRANCH}"
SRC_URI:lab3-qemuarm64 += "file://defconfig"

LINUX_VERSION:lab3-qemuarm64 ?= "5.17.5"

# the sha of the commit for 5.17.5.  git rev-list -n 1 v5.17.5
SRCREV:lab3-qemuarm64="2731bd17017d4a0e2180a1917ab22d7820a07330"

LINUX_VERSION_EXTENSION:lab3-qemuarm64 ?= "-custom"

#SRC_URI:lab3-qemuarm64 += "file://yocto-testmod.patch"
#SRC_URI:lab3-qemuarm64 += "file://lab3.cfg"
```

5.17.5 was the latest stable release when the lab material was developed

Setup our build environment

We're already in this environment from Lab#1

- `$ cd ~`
- `$ . poky/oe-init-build-env build-kernel/`
- `$ vim conf/local.conf`
- re-comment `#MACHINE = "lab2-qemuarm64"`
- uncomment `#MACHINE = "lab3-qemuarm64"`
- save and exit vim (`:wq`)
- `$ bitbake-layers remove-layer ~/kernel-lab-layers/meta-lab2-qemuarm64`
- `$ bitbake-layers add-layer ~/kernel-lab-layers/meta-lab3-qemuarm64`
- `$ cat conf/bblayers.conf`

```
    /home/<user>/kernel-lab-layers/meta-lab3-qemuarm64 \
```

Lab #3 -- Build and Boot the Image

The git clone of the kernel would have taken ~15-30 minutes. We did it for you already.

- `bitbake core-image-base` (~1.5 minutes: with the magic of sstate)
- `runqemu slirp nographic tmp/deploy/images/lab3-qemuarm64/Image-lab3-qemuarm64.bin tmp/deploy/images/lab3-qemuarm64/core-image-base-lab3-qemuarm64.ext4`
  - slirp: user space networking (no elevated privileges required)
  - nographic: run in console, do not launch a GUI window
  - path to kernel image
  - path to rootfs

Booting our linux-yocto-custom kernel

```
[    6.135776] EXT4-fs (vda): re-mounted. Quota mode: disabled.

...

Poky (Yocto Project Reference Distro) 4.0 lab3-qemuarm64 /dev/ttyAMA0

lab3-qemuarm64 login: root
root@lab3-qemuarm64:~# uname -a
Linux lab3-qemuarm64 5.17.5-custom #1 SMP PREEMPT Sat May 7 18:02:02 UTC 2022 aarch64 GNU/Linux
root@lab3-qemuarm64:~#
```

linux-yocto-custom_git.bb add patch and config

~/kernel-lab-layers/meta-lab3-qemuarm64/recipes-kernel/linux/linux-yocto-custom_git.bb

```
inherit kernel
require recipes-kernel/linux/linux-yocto.inc

# KBRANCH is the branch the used for the git clone. In this case the tip of 5.17 stable
KBRANCH = "linux-5.17.y"

SRC_URI:lab3-qemuarm64 =
"git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git;protocol=git;nocheckout=1;
branch=${KBRANCH}"
SRC_URI:lab3-qemuarm64 += "file://defconfig"

LINUX_VERSION:lab3-qemuarm64 ?= "5.17.5"

# the sha of the commit for 5.17.5.  git rev-list -n 1 v5.17.5
SRCREV:lab3-qemuarm64="2731bd17017d4a0e2180a1917ab22d7820a07330"

LINUX_VERSION_EXTENSION:lab3-qemuarm64 ?= "-custom"

SRC_URI:lab3-qemuarm64 += "file://yocto-testmod.patch"
SRC_URI:lab3-qemuarm64 += "file://lab3.cfg"
```

Uncomment these two SRC_URI lines

Lab #3 -- Rebuild (only) the Kernel

```
PREFERRED_PROVIDER:virtual/kernel = "linux-yocto-custom"
```

- **bitbake -c compile -f virtual/kernel**
  - This forces the "**do_compile**" task to be rerun.
- **bitbake -c deploy virtual/kernel**
  - This deploys the kernel image and artifacts to the **tmp/deploy/images/lab3-qemuarm64** directory.

```
MACHINE = "lab3-qemuarm64"
```

Lab #3 -- Boot the Image

"Up" arrow to reuse the command (it hasn't changed) or "Ctrl-R" and start typing for bash completion.

- `runqemu slirp nographic tmp/deploy/images/lab3-qemuarm64/Image-lab3-qemuarm64.bin tmp/deploy/images/lab3-qemuarm64/core-image-base-lab3-qemuarm64.ext4`
  - slirp: user space networking (no elevated privileges required)
  - nographic: run in console, do not launch a GUI window
  - path to kernel image
  - path to rootfs

Boot and look for module

```
Poky (Yocto Project Reference Distro) 4.0 lab3-qemuarm64 /dev/ttyAMA0

lab3-qemuarm64 login: root
root@lab3-qemuarm64:~# dmesg | grep Billroy
root@lab3-qemuarm64:~# lsmod
Module                  Size  Used by
root@lab3-qemuarm64:~# ls -al /lib/modules/5.17.5-custom/kernel/
drwxr-xr-x    3 root     root          1024 Mar  9  2018 .
drwxr-xr-x    3 root     root          1024 Mar  9  2018 ..
drwxr-xr-x    3 root     root          1024 Mar  9  2018 fs
root@lab3-qemuarm64:~#
```

Where is drivers/misc/yocto-testmod.ko?

machine/lab3-qemu.conf: add module to image

~/kernel-lab-layers/meta-lab3-qemuarm64/conf/machine/lab3-qemuarm64.conf

```
#@TYPE: Machine
#@NAME: lab3-qemuarm64

#@DESCRIPTION: Machine configuration for lab3-qemuarm64 systems

PREFERRED_PROVIDER_virtual/kernel ?= "linux-yocto-custom"

require conf/machine/include/qemu.inc
require conf/machine/include/arm/armv8a/tune-cortexa57.inc

KERNEL_IMAGETYPE = "Image"

UBOOT_MACHINE ?= "qemu_arm64_defconfig"

...

MACHINE_ESSENTIAL_EXTRA_RRECOMMENDS += "kernel-module-yocto-testmod"
```

Uncomment the MACHINE_ESSENTIAL_EXTRA_RRECOMMENDS line at the end of the file

linux-yocto-custom_git.bb deploy and autoload module

~/kernel-lab-layers/meta-lab3-qemuarm64/recipes-kernel/linux/linux-yocto-custom_git.bb

```
# KBRANCH is the branch the used for the git clone. In this case the tip of 5.17 stable
KBRANCH = "linux-5.17.y"

SRC_URI:lab3-qemuarm64 =
"git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git;protocol=git;nocheckout=1;
branch=${KBRANCH}"
SRC_URI:lab3-qemuarm64 += "file://defconfig"

LINUX_VERSION:lab3-qemuarm64 ?= "5.17.5"

# the sha of the commit for 5.17.5.  git rev-list -n 1 v5.17.5
SRCREV:lab3-qemuarm64="2731bd17017d4a0e2180a1917ab22d7820a07330"

LINUX_VERSION_EXTENSION:lab3-qemuarm64 ?= "-custom"

SRC_URI:lab3-qemuarm64 += "file://yocto-testmod.patch"
SRC_URI:lab3-qemuarm64 += "file://lab3.cfg"

KERNEL_MODULE_AUTOLOAD += "yocto-testmod"
```

Uncomment the KERNEL_MODULE_AUTOLOAD line

Fix yocto-testmod output (Try it yourself without this)

~/kernel-lab-layers/meta-lab3-qemuarm64/recipes-kernel/linux/linux-yocto-custom/yocto-testmod.patch

```
+#include <linux/module.h>
+
+static int __init yocto_testmod_init(void)
+{
+       pr_info("Billroy quacked here!\n");
+
+       return 0;
+}
+
+static void __exit yocto_testmod_exit(void)
+{
+       pr_info("Billroy did not quack here!\n");
+}
+
+module_init(yocto_testmod_init);
+module_exit(yocto_testmod_exit);
```

Terminate the string with \n

Terminate the string with \n

Lab #3 -- Rebuild and Reboot the Image

- `bitbake core-image-base`
- `runqemu slirp nographic tmp/deploy/images/lab3-qemuarm64/Image-lab3-qemuarm64.bin tmp/deploy/images/lab3-qemuarm64/core-image-base-lab3-qemuarm64.ext4`
  - slirp: user space networking (no elevated privileges required)
  - nographic: run in console, do not launch a GUI window
  - path to kernel image
  - path to rootfs

We changed the image metadata, not just the kernel this time.

```
real    10m30.378s
user    0m1.486s
sys     0m0.432s
```

Reboot and look for module again (SUCCESS!)

```
[    7.270811] EXT4-fs (vda): re-mounted. Quota mode: disabled.
[    8.064168] Billroy quacked here!
...
Poky (Yocto Project Reference Distro) 4.0 lab3-qemuarm64 /dev/ttyAMA0

lab3-qemuarm64 login: root
root@lab3-qemuarm64:~# lsmod
Module                  Size  Used by
yocto_testmod          16384  0
root@lab3-qemuarm64:~# ls -la /lib/modules/5.17.5-custom/kernel/drivers/misc/
drwxr-xr-x    2 root     root          1024 Mar  9  2018 .
drwxr-xr-x    3 root     root          1024 Mar  9  2018 ..
-rw-r--r--    1 root     root          4456 Mar  9  2018 yocto-testmod.ko
root@lab3-qemuarm64:~# cat /etc/modules-load.d/yocto-testmod.conf
yocto-testmod
root@lab3-qemuarm64:~# rmmod yocto_testmod
[   59.367837] Billroy did not quack here!
root@lab3-qemuarm64:~#
```

Without the \n fix in yocto-testmod.patch, we would not see the “Billroy” messages.

Lab#3 Complete!

Hands-on Kernel Lab #4
Exercise #1

Working with local git tree and out-of-tree module

Lab #4

- Getting to know the environment
  - ~/kernel-lab-layers/meta-lab4-qemuarm64

```
meta-lab4-qemuarm64/
├── conf
│   ├── layer.conf
│   └── machine
│       └── lab4-qemuarm64.conf
└── recipes-kernel
    ├── hello-mod
    │   ├── files
    │   │   ├── COPYING
    │   │   ├── Makefile
    │   │   └── hello.c
    │   └── hello-mod_0.1.bb
    └── linux
        ├── linux-stable-custom
        │   └── defconfig
        └── linux-stable-custom_git.bb
```

machine/lab4-qemu.conf

~/kernel-lab-layers/meta-lab4-qemuarm64/conf/machine/lab4-qemuarm64.conf

```
#@TYPE: Machine
#@NAME: lab4-qemuarm64

#@DESCRIPTION: Machine configuration for lab4-qemuarm64 systems

PREFERRED_PROVIDER_virtual/kernel ?= "linux-stable-custom"

require conf/machine/include/qemu.inc
require conf/machine/include/arm/armv8a/tune-cortexa57.inc

KERNEL_IMAGETYPE = "Image"

UBOOT_MACHINE ?= "qemu_arm64_defconfig"

...
```

linux-stable-custom_git.bb

~/kernel-lab-layers/meta-lab4-qemuarm64/recipes-kernel/linux/linux-stable-custom_git.bb

```
inherit kernel
require recipes-kernel/linux/linux-yocto.inc

SRC_URI:lab4-qemuarm64 =
"git://${HOME}/linux-stable-work.git;protocol=file;name=machine;branch=${KBRANCH}"
SRC_URI:lab4-qemuarm64 +=
"git://git.yoctoproject.org/yocto-kernel-cache;type=kmeta;name=meta;branch=yocto-5.15;destsuffix=
${KMETA}"
SRC_URI:lab4-qemuarm64 += "file://defconfig"

KBRANCH = "work-branch"
KMETA = "kernel-meta"

LINUX_VERSION:lab4-qemuarm64 ?= "5.15.37"
LINUX_VERSION_EXTENSION:lab4-qemuarm64 ?= "-custom"

SRCREV_machine:lab4-qemuarm64 = "${AUTOREV}"
SRCREV_meta:lab4-qemuarm64 = "b37a7198339ac27d27aec07ec5e952cc74c137f4"
```

Our local working git tree

5.15.37 was the latest (yocto) stable release when the lab material was developed

Commit for 5.15.37 in yocto-kernel-cache

Setup our build environment

We're already in this environment from Lab#1

- `$ cd ~`
- `$ . poky/oe-init-build-env build-kernel/`
- `$ vim conf/local.conf`
- re-comment `#MACHINE = "lab3-qemuarm64"`
- uncomment `#MACHINE = "lab4-qemuarm64"`
- save and exit vim (`:wq`)
- `$ bitbake-layers remove-layer ~/kernel-lab-layers/meta-lab3-qemuarm64`
- `$ bitbake-layers add-layer ~/kernel-lab-layers/meta-lab4-qemuarm64`
- `$ cat conf/bblayers.conf`

```
    /home/<user>/kernel-lab-layers/meta-lab4-qemuarm64 \
```

Lab #4 -- Clone our working git tree

- `pushd ~`
- `git clone -b v5.15.37 ~/DOWNLOADS/git2/git.kernel.org.pub.scm.linux.kernel.git.stable.linux-stable.git linux-stable-work.git`
- `cd linux-stable-work.git/`
- `git checkout -b work-branch`
- `popd`
- You should be in `~/build-kernel`

Re-use download already done for you.

Lab #4 -- Build and Boot the Image

- `bitbake core-image-base` (~1.5 minutes: through the magic of sstate)
- `runqemu slirp nographic tmp/deploy/images/lab4-qemuarm64/Image-lab4-qemuarm64.bin tmp/deploy/images/lab4-qemuarm64/core-image-base-lab4-qemuarm64.ext4`
  - slirp: user space networking (no elevated privileges required)
  - nographic: run in console, do not launch a GUI window
  - path to kernel image
  - path to rootfs

Booting our linux-stable-custom kernel

```
Poky (Yocto Project Reference Distro) 4.0 lab4-qemuarm64 /dev/ttyAMA0

lab4-qemuarm64 login: root
root@lab4-qemuarm64:~# uname -a
Linux lab4-qemuarm64 5.15.37-custom #1 SMP PREEMPT Sun May 1 15:22:35 UTC 2022 aarch64 GNU/Linux
root@lab4-qemuarm64:~#
```

hello-mod_0.1.bb

~/kernel-lab-layers/meta-lab4-qemuarm64/recipes-kernel/hello-mod/hello-mod_0.1.bb

```
DESCRIPTION = "hello-world-mod tests the module.bbclass mechanism."
LICENSE = "GPLv2"
LIC_FILES_CHKSUM = "file://COPYING;md5=12f884d2ae1ff87c09e5b7ccc2c4ca7e"

inherit module

PR = "r0"
PV = "0.1"

SRC_URI = "file://Makefile \
           file://hello.c \
           file://COPYING \
          "

S = "${WORKDIR}"
```

module.bbclass expects a Makefile, code and a license file

hello.c

~/kernel-lab-layers/meta-lab4-qemuarm64/recipes-kernel/hello-mod/files/hello.c

```
#include <linux/module.h>

int init_module(void)
{
        pr_info("Hello World!\n");
        return 0;
}

void cleanup_module(void)
{
        pr_info("Goodbye Cruel World!\n");
}

MODULE_LICENSE("GPL");
```

Makefile

~/kernel-lab-layers/meta-lab4-qemuarm64/recipes-kernel/hello-mod/files/Makefile

```
obj-m := hello.o

SRC := $(shell pwd)

all:
	$(MAKE) -C $(KERNEL_SRC) M=$(SRC)

modules_install:
	$(MAKE) -C $(KERNEL_SRC) M=$(SRC) modules_install

clean:
	rm -f *.o *~ core .depend .*.cmd *.ko *.mod.c
	rm -f Module.markers Module.symvers modules.order
	rm -rf .tmp_versions Modules.symvers
```

machine/lab4-qemu.conf: add module to image

~/kernel-lab-layers/meta-lab4-qemuarm64/conf/machine/lab4-qemuarm64.conf

```
#@TYPE: Machine
#@NAME: lab4-qemuarm64

#@DESCRIPTION: Machine configuration for lab4-qemuarm64 systems

PREFERRED_PROVIDER_virtual/kernel ?= "linux-stable-custom"

require conf/machine/include/qemu.inc
require conf/machine/include/arm/armv8a/tune-cortexa57.inc

KERNEL_IMAGETYPE = "Image"

UBOOT_MACHINE ?= "qemu_arm64_defconfig"

...

MACHINE_ESSENTIAL_EXTRA_RRECOMMENDS += "hello-mod"
```

Uncomment the MACHINE_ESSENTIAL_EXTRA_RRECOMMENDS line at the end of the file

not kernel-module-hello-mod because it is a standalone recipe, not built with linux-yocto-custom

Lab #4 -- Rebuild and Reboot the Image

- `bitbake core-image-base`
- `runqemu slirp nographic tmp/deploy/images/lab4-qemuarm64/Image-lab4-qemuarm64.bin tmp/deploy/images/lab4-qemuarm64/core-image-base-lab4-qemuarm64.ext4`
  - slirp: user space networking (no elevated privileges required)
  - nographic: run in console, do not launch a GUI window
  - path to kernel image
  - path to rootfs

We changed the image metadata, not the kernel this time.

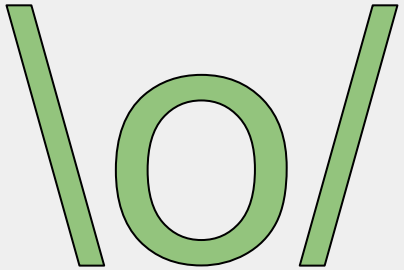
```
real    9m49.369s
user    0m1.261s
sys     0m0.422s
```

Reboot and look for module (SUCCESS!)

```
Poky (Yocto Project Reference Distro) 4.0 lab4-qemuarm64 /dev/ttyAMA0

lab4-qemuarm64 login: root
root@lab4-qemuarm64:~#  ls -la /lib/modules/5.15.37-custom/extra/
drwxr-xr-x    2 root     root          1024 Mar  9  2018 .
drwxr-xr-x    4 root     root          1024 Mar  9  2018 ..
-rw-r--r--    1 root     root          4168 Mar  9  2018  hello.ko
root@lab4-qemuarm64:~#  modprobe hello
[  100.756085] hello: loading out-of-tree module taints kernel.
[  100.770783] Hello World!
root@lab4-qemuarm64:~# lsmod
Module                  Size  Used by
hello                  16384  0

root@lab4-qemuarm64:~#  rmmod hello
[  106.346045] Goodbye Cruel World!
root@lab4-qemuarm64:~#
```



Hands-on Kernel Lab #4
Exercise #2

Modifying kernel in local git tree

Lab #4 -- Modifying our local kernel

- `$ pushd ~`
- `$ cd linux-stable-work.git/`
- `$ vim fs/filesystems.c`

fs/filesystems.c: add a pr_info statement

~/linux-stable-work.git/fs/filesystems.c

```
#ifdef CONFIG_PROC_FS
static int filesystems_proc_show(struct seq_file *m, void *v)
{
        struct file_system_type * tmp;

        read_lock(&file_systems_lock);
        tmp = file_systems;
        while (tmp) {
                seq_printf(m, "%s\t%s\n",
                        (tmp->fs_flags & FS_REQUIRES_DEV) ? "" : "nodev",
                        tmp->name);
                tmp = tmp->next;
        }
        read_unlock(&file_systems_lock);

        pr_info("Kilfoy was here!\n");

        return 0;
}
```

Insert this line

Verify the changes with `git diff -p HEAD`

```
$ git diff -p HEAD
diff --git a/fs/filesystems.c b/fs/filesystems.c
index 58b9067b2391..7fcffadaeab1 100644
--- a/fs/filesystems.c
+++ b/fs/filesystems.c
@@ -247,6 +247,9 @@ static int filesystems_proc_show(struct seq_file *m, void *v)
                tmp = tmp->next;
        }
        read_unlock(&file_systems_lock);
+
+       pr_info("Kilfoy was here!\n");
+
        return 0;
 }
```

Lab #4 -- Commit our change

- `$ git config --global user.email "you@example.com"`
- `$ git config --global user.name "Your Name"`

```
$ git commit -a -m "fs/filesystems.c: add a message
that will be logged to the kernel log when you 'cat
/proc/filesystems'."
[work-branch b685d93cfaf1] fs/filesystems.c: add a
message that will be logged to the kernel log when
you 'cat /proc/filesystems'.
 1 file changed, 3 insertions(+)
```

- `$ git log`

Verify the commit with `git log`

```
$ git log
commit b685d93cfaf1f011d75568098954904022c88c6b (HEAD -> work-branch)
Author: Your Name <you@example.com>
Date:   Wed May 11 02:47:54 2022 +0000

    fs/filesystems.c: add a message that will be logged to the kernel log when you 'cat
/proc/filesystems'.

commit 4bf7f350c1638def0caa1835ad92948c15853916 (tag: v5.15.37)
Author: Greg Kroah-Hartman <gregkh@linuxfoundation.org>
Date:   Sun May 1 17:22:35 2022 +0200

    Linux 5.15.37

    Link: https://lore.kernel.org/r/20220429104052.345760505@linuxfoundation.org
    Tested-by: Florian Fainelli <f.fainelli@gmail.com>
    Tested-by: Jon Hunter <jonathanh@nvidia.com>
    Tested-by: Shuah Khan <skhan@linuxfoundation.org>
...
```

q to quit

Lab #4 -- Build our modified kernel

- `$ popd`
- You should be in `~/build-kernel`

We need to checkout our modified source

- `$ bitbake -c cleanall virtual/kernel`
- `$ bitbake -c deploy virtual/kernel`

```
real    11m25.970s
user    0m1.101s
sys     0m0.264s
```

Lab #4 -- Boot the Image

"Up" arrow to reuse the command (it hasn't changed) or "Ctrl-R" and start typing for bash completion.

- `runqemu slirp nographic tmp/deploy/images/lab4-qemuarm64/Image-lab4-qemuarm64.bin tmp/deploy/images/lab4-qemuarm64/core-image-base-lab4-qemuarm64.ext4`
  - slirp: user space networking (no elevated privileges required)
  - nographic: run in console, do not launch a GUI window
  - path to kernel image
  - path to rootfs

Trigger our change with `cat /proc/filesystems`

```
[    7.345531] EXT4-fs (vda): re-mounted. Opts: (null). Quota mode: disabled.
[    7.515224] Kilfoy was here!
[    7.528689] Kilfoy was here!
ALSA: Restoring mixer settings...
INIT: Entering runlevel: 5
Configuring network interfaces... ip: RTNETLINK answers: File exists
Starting system message bus: dbus.
Starting rpcbind daemon...done.
Starting bluetooth: bluetoothd.
Starting syslogd/klogd: done
 * Starting Avahi mDNS/DNS-SD Daemon: avahi-daemon
   ...done.

Poky (Yocto Project Reference Distro) 4.0 lab4-qemuarm64 /dev/ttyAMA0

lab4-qemuarm64 login: root
root@lab4-qemuarm64:~#  cat /proc/filesystems
[   39.568087] Kilfoy was here!

 ...
```

every time /proc/filesystems is read by other processes we see our pr_info message

Hands-on Kernel Lab #4
Exercise #3

Using linux-yocto kernel in local git tree

Lab #4 -- Clone our working linux-yocto git tree

- `$ pushd ~`
- `$ git clone -b v5.10 ~/DOWNLOADS/git2/git.yoctoproject.org.linux-yocto.git linux-yocto-5.10.git`
- `$ cd linux-yocto-5.10.git/`
- `$ git checkout v5.10/standard/base`
- `$ popd`
- You should be in `~/build-kernel`

Re-use download already done for you.

Switch to lab2 environment

We're already in this environment from Lab#1

- `$ cd ~`
- `$ . poky/oe-init-build-env build-kernel/`
- `$ vim conf/local.conf`
- re-comment `#MACHINE = "lab4-qemuarm64"`
- uncomment `#MACHINE = "lab2-qemuarm64"`
- save and exit vim (`:wq`)
- `$ bitbake-layers remove-layer ~/kernel-lab-layers/meta-lab4-qemuarm64`
- `$ bitbake-layers add-layer ~/kernel-lab-layers/meta-lab2-qemuarm64`
- `$ cat conf/bblayers.conf`

```
  /home/<user>/kernel-lab-layers/meta-lab2-qemuarm64 \
```

linux-yocto_5.10.bbappend: point to our local git tree

~/kernel-lab-layers/meta-lab2-qemuarm64/recipes-kernel/linux/linux-yocto_5.10.bbappend

```
FILESEXTRAPATHS:prepend := "${THISDIR}/files:"

COMPATIBLE_MACHINE:lab2-qemuarm64 = "lab2-qemuarm64"

SRC_URI = "git://${HOME}/linux-yocto-5.10.git;protocol=file;name=machine;branch=${KBRANCH}; \

git://git.yoctoproject.org/yocto-kernel-cache;type=kmeta;name=meta;branch=yocto-5.10;destsuffix=${KMETA}"
KERNEL_VERSION_SANITY_SKIP="1"

SRCREV_machine:pn-linux-yocto:lab2-qemuarm64 ?= "${AUTOREV}"
SRCREV_meta:pn-linux-yocto:lab2-qemuarm64 ?= "${AUTOREV}"

KBRANCH:lab2-qemuarm64 = "v5.10/standard/base"
KMACHINE:lab2-qemuarm64  = "qemuarm64"

KERNEL_FEATURES:append:lab2-qemuarm64 = " cfg/smp.scc"

SRC_URI += "file://mtd-block.cfg"

#SRC_URI += "file://yocto-testmod.patch"
#SRC_URI += "file://lab2.cfg"
```

machine/lab2-qemu.conf: verify 5.10 LTS kernel

~/kernel-lab-layers/meta-lab2-qemuarm64/conf/machine/lab2-qemuarm64.conf

```
#@TYPE: Machine
#@NAME: lab2-qemuarm64

#@DESCRIPTION: Machine configuration for lab2-qemuarm64 systems

PREFERRED_PROVIDER_virtual/kernel ?= "linux-yocto"
#PREFERRED_VERSION_linux-yocto ?= "5.15%"
PREFERRED_VERSION_linux-yocto ?= "5.10%"


require conf/machine/include/qemu.inc
require conf/machine/include/arm/armv8a/tune-cortexa57.inc

KERNEL_IMAGETYPE = "Image"

UBOOT_MACHINE ?= "qemu_arm64_defconfig"
...
```

Rebuild the Kernel and Reboot the Image

```
real    11m4.820s
user    0m1.038s
sys     0m0.231s
```

- `bitbake -c deploy virtual/kernel`
- `runqemu slirp nographic tmp/deploy/images/lab2-qemuarm64/Image-lab2-qemuarm64.bin tmp/deploy/images/lab2-qemuarm64/core-image-base-lab2-qemuarm64.ext4`
  - slirp: user space networking (no elevated privileges required)
  - nographic: run in console, do not launch a GUI window
  - path to kernel image
  - path to rootfs

Boot our local linux-yocto LTS Kernel

```
Poky (Yocto Project Reference Distro) 4.0 lab2-qemuarm64 /dev/ttyAMA0

lab2-qemuarm64 login: root
root@lab2-qemuarm64:~# uname -a
Linux lab2-qemuarm64 5.10.114-yocto-standard #1 SMP PREEMPT Mon May 9 20:36:56 UTC 2022 aarch64
GNU/Linux
root@lab2-qemuarm64:~#
```

Hands-on Kernel Lab #4
Exercise #4

Modifying linux-yocto kernel in local git tree

Lab #4 -- Modifying our local linux-yocto kernel

- `$ pushd ~`
- `$ cd linux-yocto-5.10.git/`
- `$ vim fs/filesystems.c`

fs/filesystems.c: add a printk statement

~/linux-yocto-5.10.git/fs/filesystems.c

```
#ifdef CONFIG_PROC_FS
static int filesystems_proc_show(struct seq_file *m, void *v)
{
        struct file_system_type * tmp;

        read_lock(&file_systems_lock);
        tmp = file_systems;
        while (tmp) {
                seq_printf(m, "%s\t%s\n",
                        (tmp->fs_flags & FS_REQUIRES_DEV) ? "" : "nodev",
                        tmp->name);
                tmp = tmp->next;
        }
        read_unlock(&file_systems_lock);

        pr_info("Robroy drank here!\n");

        return 0;
}
```

Search quickly with /`filesystems_proc_show(`

Insert this line

Verify the changes with `git diff -p HEAD`

```
$ git diff -p HEAD
diff --git a/fs/filesystems.c b/fs/filesystems.c
index 90b8d879fbaf..81da0acf2a2c 100644
--- a/fs/filesystems.c
+++ b/fs/filesystems.c
@@ -240,6 +240,9 @@ static int filesystems_proc_show(struct seq_file *m, void *v)
                tmp = tmp->next;
        }
        read_unlock(&file_systems_lock);
+
+        pr_info("Robroy drank here!\n");
+
        return 0;
 }
```

Commit our change

- `$ git config --global user.email "you@example.com"`
- `$ git config --global user.name "Your Name"`
- `$ git commit -a -s -F- << EOF`

```
fs/filesystems.c: add debug

DO NOT MERGE

add a message that will be logged to the kernel log when you “cat
/proc/filesystems”.
EOF
[v5.10/standard/base e4788432a83c] fs/filesystems.c: add debug
 1 file changed, 3 insertions(+)
```

- `$ git log`

Verify the commit with `git log v5.10/standard/base`

```
commit e4788432a83c46d3fbfd6d23e129c82e13b95fd4 (HEAD -> v5.10/standard/base)
Author: Your Name <you@example.com>
Date:   Wed May 11 16:51:12 2022 +0000

    fs/filesystems.c: add debug

    DO NOT MERGE

    add a message that will be logged to the kernel log when you "cat /proc/filesystems".

    Signed-off-by: Your Name <you@example.com>

commit 7b27bcac98c4b1fdff396ecd50ea8d5875ed26b8
Author: Greg Kroah-Hartman <gregkh@linuxfoundation.org>
Date:   Wed Apr 27 13:53:58 2022 +0200

    Linux 5.10.113

    Link: https://lore.kernel.org/r/20220426081741.202366502@linuxfoundation.org
    Tested-by: Jon Hunter <jonathanh@nvidia.com>
...
    Signed-off-by: Greg Kroah-Hartman <gregkh@linuxfoundation.org>
```

q to quit

Build our modified kernel

- `$ popd`
- You should be in `~/build-kernel`

We need to checkout our modified source

- `$ bitbake -c cleanall virtual/kernel`
- `$ bitbake -c deploy virtual/kernel`

```
real    11m8.152s
user    0m1.090s
sys     0m0.233s
```

NOTE: If the syntax of the **SRCREV ${AUTOREV}** statements is correct, you should see the commit hash from our change in the build:

```
$ ls tmp/work/lab2_qemuarm64-poky-linux/linux-yocto/
   5.10.109+gitAUTOINC+b368b4c1c8_e4788432a8-r0/
```

Reboot the Image

- `runqemu slirp nographic tmp/deploy/images/lab2-qemuarm64/Image-lab2-qemuarm64.bin tmp/deploy/images/lab2-qemuarm64/core-image-base-lab2-qemuarm64.ext4`
  - slirp: user space networking (no elevated privileges required)
  - nographic: run in console, do not launch a GUI window
  - path to kernel image
  - path to rootfs

Trigger our change with `cat /proc/filesystems`

```
[    5.096484] Robroy drank here!
[    5.098315] Robroy drank here!
Starting udev
[    6.000457] udevd[134]: starting version 3.2.10
[    6.180141] udevd[135]: starting eudev-3.2.10
[    7.141875] EXT4-fs (vda): re-mounted. Opts: (null)
[    7.528190] Robroy drank here!
ALSA: Restoring mixer settings...
...

Poky (Yocto Project Reference Distro) 4.0 lab2-qemuarm64 /dev/ttyAMA0

lab2-qemuarm64 login: root
root@lab2-qemuarm64:~# cat /proc/filesystems
[   27.037486] Robroy drank here!
nodev   sysfs
nodev   tmpfs
...
root@lab2-qemuarm64:~#
```

every time /proc/filesystems is read by other processes we see our printk message

Lab#4 Complete!

Achievement Unlocked!

yocto PROJECT | THE LINUX FOUNDATION

What is the Yocto Project® ?

IT'S NOT AN EMBEDDED LINUX DISTRIBUTION, IT CREATES A CUSTOM ONE FOR YOU.

The Yocto Project (YP) is an open source collaboration project that helps developers create custom Linux-based systems regardless of the hardware architecture.

The project provides a flexible set of tools and a space where embedded developers worldwide can share technologies, software stacks, configurations, and best practices that can be used to create tailored Linux images for embedded and IOT devices, or anywhere a customized Linux OS is needed.

yocto PROJECT | THE LINUX FOUNDATION